More Instant Messaging Interoperability
Internet-Draft
Intended status: Informational
Expires: 15 August 2025

R. Mahy
Rohan Mahy Consulting Service

11 February 2025

Application State Components for More Instant Messaging Interoperability (MIMI)

draft-mahy-mimi-app-components-01

Abstract

This document presents structures for room metadata, participant lists, pre-authorized roles for future participants, and role-based access control, all of which are intended for use in MIMI (More Instant Messaging Interoperability).

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://rohanmahy.github.io/mimi-app-components/draft-mahy-mimi-app-components.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-mahy-mimi-app-components/.

Discussion of this document takes place on the More Instant Messaging Interoperability Working Group mailing list (mailto:mimi@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/mimi/. Subscribe at https://www.ietf.org/mailman/listinfo/mimi/.

Source for this draft and an issue tracker can be found at https://github.com/rohanmahy/mimi-app-components.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 15 August 2025.

Table of Contents

1. Introduction

This document introduces specific structures to carry room metadata, the room participant list (introduced conceptually in [I-D.ietf-mimi-arch]), room policy for preauthorizing users into the room based on identity-based attributes, and per-room role definitions. Each of these structures is represented as an MLS application component as defined in [I-D.barnes-mls-appsync] (soon to be merged into [I-D.ietf-mls-extensions]). Each component is represented in the MLS GroupContext for the room.

This document is provided as a standalone document as it is short and easy to receive early review in its short format. The goal is to incorporate the contents of this draft into [I-D.ietf-mimi-room-policy].

While this work is intended for use with MIMI, it is suitable for use with other systems using MLS (for example, non-authority-based messaging systems) that require similar functionality.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Room Metadata

The Room Metadata component contains data about a room which might be displayed as human-readable information for the room, such as the name of the room and a URL pointing to its room image/avatar.

It can contain a list of room_descriptions, each of which can have a specific language_tag and media_type along with the description_content. An empty media_type implies text/plain;charset=utf-8.

RoomMetaData is the format of the data field inside the ComponentData struct for the Room Metadata component in the application_data GroupContext extension.

```
/* a valid URI (ex: MIMI URI) */
struct {
  opaque uri<V>;
} Uri;

/* a sequence of valid UTF8 without nulls */
struct {
  opaque string<V>;
} UTF8String;

struct {
  /* an empty media_type is equivalent to text/plain;charset=utf-8 */
  opaque media_type<V>;
  opaque language_tag<V>;
  opaque description_content<V>;
} RichDescription;

struct {
  Uri room_uri;
  UTF8String room_name;
  RichDescription room_descriptions<V>;
  /* an https URI resolving to an avatar image */
  Uri room_avatar;
  UTF8String room_subject;
  UTF8String room_mood;
} RoomMetaData;

RoomMetaData RoomMetaUpdate;
```

RoomMetaUpdate (which has the same format as RoomMetaData) is the format of the update field inside the AppDataUpdate struct in an AppDataUpdate Proposal for the Room Metadata component. If the contents of the update field are valid and if the proposer is authorized to generate such an update, the value of the update field completely replaces the value of the data field.

Only a single Room metadata update is valid per commit.

4. Participant List

The participant list is a list of "users" in a room. Within a room, each user is assigned exactly one _role_ (expressed with a role_index and described in Section 6) at any given time (specifically within any MLS epoch). In a room that has multiple MLS clients per "user", the identifier for each user in participants.user is the same across all that user’s clients in the room. Note that each user has a single role at any point in time, and therefore all clients of the same user also have the same role.

The participant list may include inactive participants, which currently do not have any clients in the corresponding MLS group, for example if their clients do not have available KeyPackages or if all of their clients are temporarily "kicked" out of the group. The participant list can also contain participants that are explicitly banned, by assigning them a suitable role which does not have any capabilities.

```
struct {
  opaque user<V>;
  uint32 role_index;
} UserRolePair;

struct {
  UserRolePair participants<V>;
} ParticipantListData;
```

ParticipantListData is the format of the data field inside the ComponentData struct for the Participant list Metadata component in the application_data GroupContext extension.

```
struct {
  uint32 user_index;
  uint32 role_index;
} UserindexRolePair;

struct {
  UserindexRolePair changedRoleParticipants<V>
  uint32 removedIndices<V>;
  UserRolePair addedParticipants<V>;
} ParticipantListUpdate;
```

ParticipantListUpdate is the contents of an AppDataUpdate Proposal with the component ID for the participant list. The index of the participants vector in the current ParticipantListData struct is referenced as the user_index when making changes. First the changedRoleParticipants list contains UserindexRolePairs with the index of a user who changed roles and their new role. Next is the removedIndices list which has a list of users to remove completely from the participant list. Finally there is a list of addedParticipants (which contains a user and role) that is appended to the end of the ParticipantListData.

Each of these actions (modifying a user’s role, removing a user, and adding a user) is authorized separately according to the rules specified in Section 7.1. If all the changes are authorized, the ParticipantListData is modified accordingly.

A single commit is not valid if it contain any combination of Participant list updates that operate on (add, remove, or change the role of) the same user in the participant list more than once.

5. Preauthorized Users

Preauthorized users are MIMI users and external senders that have authorization to adopt a role in a room by virtue of certain credential claims or properties, as opposed to being individually enumerated in the participant list. For example, a room for employee benefits might be available to join with the regular participant role to all full-time employees with a residence in a specific country; while anyone working in the human resources department might be able to join the same room as a moderator. This data structure is consulted in two situations: for external joins (external commits) and external proposals when the requester does not already appear in the participant list; and separately when an existing participant explicitly tries to change its _own_ role.

> Only consulting Preauthorized users in these cases prevents several attacks. For example, it prevents an explicitly banned user from rejoining a group based on a preauthorization.

PreAuthData is the format of the data field inside the ComponentData struct for the Preauthorized Participants component in the application_data GroupContext extension.

The individual PreAuthRoleEntry rules in PreAuthData are consulted one at a time. A PreAuthRoleEntry matches for a requester when every Claim.claim_id has a corresponding claim in the requester’s MLS Credential which exactly matches the corresponding claim_value. When the rules in a Preauthorized users struct match multiple roles, the requesting client receives the first role which matches its claims.

```
struct {
  /* MLS Credential Type of the "claim"  */
  CredentialType credential_type;
  /* the binary representation of an X.509 OID, a JWT claim name  */
  /* string, or the CBOR map claim key in a CWT (an int or tstr)  */
  opaque id<V>;
} ClaimId;

struct {
  ClaimId claim_id;
  opaque claim_value<V>;
} Claim;

struct {
  /* when all claims in the claimset are satisfied, the claimset */
  */ is satisfied */
  Claim claimset<V>;
  Role target_role;
} PreAuthRoleEntry;

struct {
  PreAuthRoleEntry preauthorized_entries<V>;
} PreAuthData;

PreAuthData PreAuthUpdate;
```

PreAuthUpdate (which has the same format as PreAuthData) is the format of the update field inside the AppDataUpdate struct in an AppDataUpdate Proposal for the Preauthorized Participants component. If the contents of the update field are valid and if the proposer is authorized to generate such an update, the value of the update field completely replaces the value of the data field.

> As with the definition of roles, in MIMI it is not expected that the definition of Preauthorized users would change frequently. Instead the claims in the underlying credentials would be modified without modifying the preauthorization policy.

Changing Preauthorized user definitions is sufficiently disruptive, that an update to this component is not valid if it appears in the same commit as any Participant List change, except for user removals.

Because the Preauthorized users component usually authorizes non-members, it is also a natural choice for providing concrete authorization for policy enforcing systems incorporated into or which run in coordination with the MIMI Hub provider or specific MLS Distribution Services. For example, a preauthorized role could allow the Hub to remove participants and to ban them, but not to add any users or devices. This unifies the authorization model for members and non-members.

6. Role-Based Access Control

The Role-Based Access Control component contains a list of all the roles in the room, and the capabilities associated with them. It contains a role_index, which is used to refer to the role elsewhere. (Note that role indexes might not be contiguous.) The role_index zero is reserved to refer to a participant that does not (yet) or no longer appears (or will no longer appear) in the participant list.

The component also contains a role_name (a human-readable text string name for the role), and a role_description (another string, which can have zero length).

Each Role also can contain constraints on the minimum and maximum number of participants, and the minimum and maximum number of active participants. If the minimum number is zero, there is no minimum number of participants for that particular role. If there is no maximum number of participants for a particular role, that parameter is absent.

> If the maximum number of active participants is zero, then no participants are allowed to have clients in the room’s MLS group.

The authorized_role_changes field is used to provide fine-grained control about which transitions are allowed when adding and removing participants and when moving participants to new roles, including banning/unbanning, and promoting/demoting to or from roles with moderator or administrator privileges. A more detailed discussion is in the description of the specific capabilities in the next section.

> This design results in each participant only having a single role at a time, with a single list of capabilities and an explicit list of allowed role transitions. It makes the authorization process for a verifier consistent regardless of the complexity of the set of authorization rules.

Some examples are provided in Appendix A.

RoleData is the format of the data field inside the ComponentData struct for the Role-Based Access Control component in the application_data GroupContext extension.

```
/* See MIMI Capability Types IANA registry */
uint16 CapablityType;

struct {
   uint32 from_role_index;
   uint32 target_role_indexes<V>;
} SingleSourceRoleChangeTargets;

struct {
  uint32 role_index;
  opaque role_name<V>;
  opaque role_description<V>;
  CapabilityType role_capabilities<V>;
  uint32 minimum_participants_constraint;
  optional uint32 maximum_participants_constraint;
  uint32 minimum_active_participants_constraint;
  optional uint32 maximum_active_participants_constraint;
  SingleSourceRoleChangeTargets authorized_role_changes<V>;
} Role;

struct {
  Role roles<V>;
} RoleData;

RoleData RoleUpdate;
```

RoleUpdate (which has the same format as RoleData) is the format of the update field inside the AppDataUpdate struct in an AppDataUpdate Proposal for the Role-Based Access Control component. If the contents of the update field are valid and if the proposer is authorized to generate such an update, the value of the update field completely replaces the value of the data field.

> Note that in the MIMI environment, changing the definitions of roles is anticipated to be very rare over the lifetime of a room (for example changing a room which has grown dramatically from cooperatively managed by all participants to explicitly moderated or administered).

Changing Role definitions is sufficiently disruptive, that an update to this component is not valid if it appear in the same commit as any Participant List change.

7. Role Capabilities

As described in the previous section, each role has a list of capabilities, which in rare cases could be empty. When we say that the holder of a capability can take some action, we mean that whatever entity is taking the action (a participant, a potential future participant, or an external party) has a specific entry in the Participant List struct and a corresponding role--or is preauthorized to take action with a specific role via the Preauthorized Users struct--and that the role_capabilities list contains the relevant capability.

Unless otherwise specified, capabilities apply both to sending a set of consistent MLS proposals that could be committed by any member of the corresponding MLS group, and to sending an MLS commit containing a set of consistent MLS proposals.

7.1. Membership Capabilities

The membership capabilities below allow authorized holders to update the Participant list, or change the active participants (by removing and adding MLS clients corresponding to those participants), or both.

* canAddParticipant - the holder of this capability can add another user, that is not already in the participant list, to the participant list. (This capability does not apply to the holder adding itself.) The authorized_role_changes list in the holder's role is consulted to authorize the added user's target role. The authorized_role_changes list MUST have an entry where the authorized_role_changes.from_role_index equals zero, and that entry's target_role_indexes list includes the target role. The proposed action is only authorized if the action respects both the maximum_participants_constraint (if present) and maximum_active_participants_constraint (if present) for the added user's target role. When the participant list addition for the target role is authorized, the holder is also authorized to add any MLS clients matching the added user to the room's MLS group .

* canAddOwnClient - a holder of this capability that is in the participant list, can add its own client (via an external commit or external proposal); and can add other clients that share the same user identity (via Add proposals) if the holder's client is already a member of the corresponding MLS group.

* canAddSelf - the holder of this capability can use an external commit or external proposal to add itself to the participant list. (The holder MUST NOT already appear in the participant list). Its usage differs slightly based on in which role it appears.

  - When canAddSelf appears on role zero, any user who is not already in the participant list can add itself, with certain provisions. The holder consults the authorized_role_changes list for an entry with from_role_index equal to zero. The holder can add itself with any non-zero target_role_indexes from that entry, if the action respects both the maximum_participants_constraint (if present) and maximum_active_participants_constraint (if present) for the added user's target role.

  - When canAddSelf appears on a non-zero role, a client can only become the holder of this capability via the Preauthorized users mechanism. The authorized_role_changes list in the target role MUST have an entry where the from_role_index is zero and the target_role_indexes contains the target role. In addition, the action MUST respect both the maximum_participants_constraint (if present) and maximum_active_participants_constraint (if present) for the added user's target role.

* canUseJoinCode - the holder of this capability can externally join a room using a join code for that room, provided the join code is valid, the join code refers to a valid target role, and both the maximum_participants_constraint (if present) and maximum_active_participants_constraint (if present) constraints are respected.

* canRemoveParticipant - the holder of this capability can propose a) the removal of another user (excluding itself) from the participant list, and b) removal of all of that user's clients, as a single action. There MUST NOT be any clients of the removed user in the MLS group after the corresponding commit. A proposer holding this capability consults its role's authorized_role_changes entries for an entry where from_role_index matches the target user's current role; if the target_role_indexes for that entry contains zero, and the minimum_participants_constraint and minimum_active_participants_constraint are satisfied, the proposal is authorized.

* canRemoveOwnClient - the holder of this capability can propose to remove its own client using an MLS Remove or SelfRemove proposal without changing the Participant list. Due to restrictions in MLS which insure the consistency of the group, this action cannot be committed by the leaving user. If the minimum_active_participants_constraint is satisfied, the proposal is authorized.

* canRemoveSelf - the holder of this capability can propose to remove itself from (i.e. leave) the participant list; it MUST simultaneously propose to remove all of its remaining clients from the corresponding MLS group. Due to restrictions in MLS which insure the consistency of the group, this action cannot be committed by the leaving user. A proposer holding this capability consults its role's authorized_role_changes entries for an entry where from_role_index matches its current role; if the target_role_indexes for that entry contains zero, and the minimum_participants_constraint and minimum_active_participants_constraint are satisfied, the proposal is authorized.

* canKick - the holder of this capability can propose removal of another participant's clients, without changing the Participant List. If the minimum_active_participants_constraint is satisfied, the proposal is authorized.

* canChangeUserRole - the holder of this capability is authorized to change the role of another participant (but not itself), according to the holder's authorized_role_changes list, from a role represented by an entry where the target's current role matches from_role_index to any of the non-zero target_role_indexes in the same element of authorized_role_changes. The minimum_participants_constraint and minimum_active_participants_constraint for the target user's current role, and the maximum_participants_constraint (if present) and maximum_active_participants_constraint (if present) for the target user's target role must also be satisfied.

* canChangeOwnRole - the holder of this capability is authorized to change its own role to the first non-zero role it matches in the Preauthorized users component (see Section 5). The authorized_role_changes list is _not_ consulted. The minimum_participants_constraint and minimum_active_participants_constraint for the holder's original role, and the maximum_participants_constraint (if present) and maximum_active_participants_constraint (if present) for the holder's target role must also be satisfied.

* canBan - the holder of this capability can propose to "ban" another user. Specifically, a successful ban changes the target user's role to a special "banned" role (if it exists), and removes all the banned user's clients. The "banned" role always has role_index = 1 and role_name = "banned" (without quotes).

A "banned" role does not have to exist in a room, but to use the canBan and canUnban capabilities, the role needs to exist exactly as described above. While holding canChangeUserRole and canKick capabilities would allow the same action, it could potentially allow the holder other actions which might be undesirable in some contexts, such as kicking clients without banning.

A proposer holding this capability consults its role's authorized_role_changes entries for an entry where from_role_index matches the target user's current role; if the target_role_indexes for that entry contains the role_index 1; that role_name = "banned" for the role with role_index = 1, and the minimum_participants_constraint and minimum_active_participants_constraint are satisfied, the proposal is authorized.

* canUnban - the holder of this capability can propose to "unban" another user. Specifically, a successful unban changes the target user's role from role_index = 1 to another non-zero role_index allowed by the holder's authorized_role_changes list. Adding clients for that unbanned user is _not_ authorized by this capability. The authorization of this capability is identical to the canChangeUserRole capability, except that the from_role_index for the unbanned user MUST be 1, and the role_name of role 1 MUST be "banned".

7.2. Adjust metadata

The holder of each of the following capabilities is authorized to update the Room metadata, changing the relevant field:

* canChangeRoomName

* canChangeRoomDescription

* canChangeRoomAvatar

* canChangeRoomSubject

* canChangeRoomMood

7.3. Message Capabilities

The capabilities below refer to functionality related to the instant messages, for example sent using the MIMI content format [I-D.ietf-mimi-content].

* canSendMessage - the holder can send instant messages to the room. Setting specific message fields may require additional capabilities.

* canReceiveMessage - the holder can receive instant messages from the room.

* canCopyMessage - the holder can copy content from a received instant message.

* canReportAbuse - the holder can report a franked instant message as abusive.

* canReplyToMessage - the holder can send a message replying to another message.

* canReactToMessage - the holder can send a reaction, replying to another message, and using the "reaction" disposition.

* canDeleteOwnReaction - the holder can retract (unlike) it own previous reaction.

* canDeleteOtherReaction - the holder can delete the reaction of another user’s previous reaction

* canEditOwnMessage - the holder can edit the content of one of its own previously sent messages

* canDeleteOwnMessage - the holder can retract one of its own previously sent messages

* canDeleteOtherMessage - the holder can retract messages for other users.

* canStartTopic - the holder can set the topic for a message

* canReplyInTopic - the holder can send a message replying to a previous message, using the same topic as the original sender.

* canEditOwnTopic - the holder can change the topic of a previously sent message

* canEditOtherTopic - the holder can change the topic of a message previously sent by another user.

* canSendLink - the holder can send an inline link

* canSendLinkPreview - the holder can send an inline link with an associated preview.

* canFollowLink - the holder can open a sent inline link.

* canCopyLink - the holder can copy the URL of a sent inline link.

The Hub can enforce whether a member can send a message. It can also withhold fanout of application messages to clients of a user. The other capabilities in this section can only be enforced by other clients.

7.4. Asset Capabilities

* canUploadAttachment - the holder can upload a file with the "attachent" disposition.

* canDownloadAttachment - the holder can download a file with the "attachent" disposition.

* canUploadImage - the holder can upload a file with the media type of "image" and the disposition of "render"

* canDownloadImage - the holder can download a file with the media type of "image" and the disposition of "render"

* canUploadVideo - the holder can upload a file with the media type of "video" and the disposition of "render"

* canDownloadVideo - the holder can download a file with the media type of "video" and the disposition of "render"

* canUploadSound - the holder can upload a file with the media type of "audio" and the disposition of "render"

* canDownloadSound - the holder can download a file with the media type of "audio" and the disposition of "render"

7.5. Real-time media

The MIMI Working has not yet defined requirements for real-time media, however the capabilities below are widely representative of the permissions that would be required.

* canStartCall - the holder can initiate a new real-time call/ conference

* canJoinCall - the holder can join an existing real-time call/ conference

* canSendAudio - the holder is authorized to contribute audio in a call/conference.

* canReceiveAudio - the holder is authorized to receive audio in a call/conference.

* canSendVideo - the holder is authorized to contribute video in a call/conference.

* canReceiveVideo - the holder is authorized to receive video in a call/conference.

* canShareScreen - the holder is authorized to contribute screen sharing in a call/conference

* canViewSharedScreen - the holder is authorized to receive screen sharing in a call/conference

7.6. Disruptive Policy Changes

* canChangeRoomMembershipStyle - the holder is authorized to modify the base room membership style.

* canChangeRoleDefinitions - the holder is authorized to make changes to the definitions of the Roles component.

* canChangePreauthorizedUserList - the holder is authorized to make changes to the Preauthorized Users component.

* canDestroyRoom - the holder is authorized to completely destroy the room.

* canReinitGroup - the holder is authorized to send an MLS ReInit proposal.

7.7. Reserved Capabilities

The following capability names are reserved for possible future use

* canCreateJoinCode

* canKnock

* canAcceptKnock

* canCreateSubgroup
* canSendDirectMessage
* canTargetMessage
* canChangeOwnName
* canChangeOwnPresence
* canChangeOwnMood
* canChangeOwnAvatar
* canCreateRoom
* canChangeMlsOperationalPolicies
* canChangeOtherPolicyAttribute
* MLS specific
  - update - update policy
  - PSK - psk policy
  - external proposal - general operational policy rules
  - external commit - general operational policy rules

8. Security Considerations

TODO Security

9. IANA Considerations

9.1. New MIMI Role Capabilities registry

Create a new registry with the following values assigned sequentially using the reference RFCXXXX.

canAddParticipant
canRemoveParticipant
canAddOwnClient
canRemoveSelf
canAddSelf
canCreateJoinCode - reserved for future use
canUseJoinCode

canBan
canUnBan
canKick
canKnock
canAcceptKnock
canChangeUserRole
canChangeOwnRole
canCreateSubgroup
canSendMessage
canReceiveMessage
canCopyMessage
canReportAbuse
canReactToMessage
canEditReaction
canDeleteReaction
canEditOwnMessage
canDeleteOwnMessage
canDeleteAnyMessage
canStartTopic
canReplyInTopic
canEditTopic
canSendDirectMessage
canTargetMessage
canUploadImage
canUploadVideo
canUploadAttachment
canDownloadImage
canDownloadVideo
canDownloadAttachment
canSendLink
canSendLinkPreview
canFollowLink
canCopyLink
canChangeRoomName
canChangeRoomDescription
canChangeRoomAvatar
canChangeRoomSubject
canChangeRoomMood
canChangeOwnName
canChangeOwnPresence
canChangeOwnMood
canChangeOwnAvatar
canStartCall
canJoinCall
canSendAudio
canReceiveAudio
canSendVideo
canReceiveVideo

canShareScreen
canViewSharedScreen
canChangeRoomMembershipStyle
canChangeRoleDefinitions
canChangePreauthorizedUserList
canChangeMlsOperationalPolicies
canDestroyRoom
canSendMLSReinitProposal

10. Normative References

[I-D.barnes-mls-appsync]
Jo谷l, Barnes, R., Mahy, R., and M. Mularczyk, "A Safe Application Interface to Messaging Layer Security", Work in Progress, Internet-Draft, draft-barnes-mls-appsync-01, 12 December 2024, <https://datatracker.ietf.org/doc/html/draft-barnes-mls-appsync-01>.

[I-D.ietf-mimi-arch]
Barnes, R., "An Architecture for More Instant Messaging Interoperability (MIMI)", Work in Progress, Internet-Draft, draft-ietf-mimi-arch-01, 21 November 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-mimi-arch-01>.

[I-D.ietf-mimi-content]
Mahy, R., "More Instant Messaging Interoperability (MIMI) message content", Work in Progress, Internet-Draft, draft-ietf-mimi-content-05, 20 December 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-mimi-content-05>.

[I-D.ietf-mimi-room-policy]
Mahy, R., "Room Policy for the More Instant Messaging Interoperability (MIMI) Protocol", Work in Progress, Internet-Draft, draft-ietf-mimi-room-policy-00, 15 November 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-mimi-room-policy-00>.

[I-D.ietf-mls-extensions]
Robert, R., "The Messaging Layer Security (MLS) Extensions", Work in Progress, Internet-Draft, draft-ietf-mls-extensions-05, 21 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-mls-extensions-05>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

Appendix A. Role examples

A.1. Cooperatively administered room

This is an example set of role policies, which is suitable for friends and family rooms and small groups of peers in a workgroup or club.

* no_role
  - role_index = 0
  - no capabilities
  - constraints
    - minimum_participants_constraint = 0
    - maximum_participants_constraint = null
    - minimum_active_participants_constraint = 0
    - maximum_active_participants_constraint = 0
    - authorized_role_changes = []
* banned
  - role_index = 1
  - no capabilities
  - constraints
    - minimum_participants_constraint = 0
    - maximum_participants_constraint = null
    - minimum_active_participants_constraint = 0

        - maximum_active_participants_constraint = 0
        - authorized_role_changes = []
    * ordinary_user
        - role_index = 2
        - authorized capabilities
            - canAddParticipant
            - canRemoveParticipant
            - canAddOwnClient
            - canRemoveOwnClient
            - canRemoveSelf
            - canSendMessage
            - canReceiveMessage
            - canCopyMessage
            - canReportAbuse
            - canReplyToMessage
            - canReactToMessage
            - canDeleteOwnReaction
            - canEditOwnMessage
            - canDeleteOwnMessage
            - canStartTopic
            - canReplyInTopic
            - canEditOwnTopic
            - canUploadImage
            - canUploadVideo

- o canUploadSound
- o canUploadAttachment
- o canDownloadImage
- o canDownloadVideo
- o canDownloadSound
- o canDownloadAttachment
- o canSendLink
- o canSendLinkPreview
- o canFollowLink
- o canCopyLink
- o canChangeRoomName
- o canChangeRoomAvatar
- o canChangeRoomSubject
- o canChangeRoomMood
- o canChangeOwnName
- o canChangeOwnPresence
- o canChangeOwnMood
- o canChangeOwnAvatar

- constraints
  - o minimum_participants_constraint = 0
  - o maximum_participants_constraint = null
  - o minimum_active_participants_constraint = 0
  - o maximum_active_participants_constraint = null
  - o authorized_role_changes = [(0,[2]), (2,[0])]

* group_admin
  - role_index = 3
  - authorized capabilities
    - (include all the capabilities authorized for an ordinary_user)
    - canBan
    - canUnBan
    - canKick
    - canRevokeVoice
    - canGrantVoice
    - canChangeUserRole
    - canDeleteOtherMessage
    - canEditOtherTopic
    - canChangeRoomDescription
  - constraints
    - minimum_participants_constraint = 1
    - maximum_participants_constraint = null
    - minimum_active_participants_constraint = 0
    - maximum_active_participants_constraint = null
    - authorized_role_changes = [(0,[1,2,3]), (1,[0,2,3]), (2,[0,1,3]), (3,[0,1,2])]
* super_admin
  - role_index = 4
  - authorized capabilities
    - (include all the capabilities authorized for a group_admin)

        - canChangeRoomMembershipStyle
        - canChangePreauthorizedUserList
        - canDestroyRoom
    - constraints
        - minimum_participants_constraint = 0
        - maximum_participants_constraint = null
        - minimum_active_participants_constraint = 0
        - maximum_active_participants_constraint = null
        - authorized_role_changes = [(0,[1,2,3,4]), (1,[0,2,3,4]), (2,[0,1,3,4]), (3,[0,1,2,4]), (4,[0,1,2,3])]

* policy_enforcer
    - role_index = 5
    - capabilities
        - (does not include any other capabilities)
        - canRemoveParticipant
        - canChangeUserRole
        - canBan
        - canUnban
        - canChangeRoleDefinitions
        - canChangePreauthorizedUserList
        - canChangeMlsOperationalPolicies
        - canDestroyRoom
        - canSendMLSReinitProposal
    - constraints
        - minimum_participants_constraint = 1

        - maximum_participants_constraint = 2
        - minimum_active_participants_constraint = 0
        - maximum_active_participants_constraint = 0
        - authorized_role_changes = [(0,[1]), (1,[0]), (2,[0,1]), (3,[0,1]), (4,[0,1])]
    - Notes: can remove a banned user from the list (cleanup) but not restore them

A.2. Strictly administered room

This is an example set of role policies, which is suitable for friends and family rooms and small groups of peers in a workgroup or club.

* no_role
    - role_index = 0
    - authorized capabilities
        - canUseJoinCode
    - constraints
        - minimum_participants_constraint = 0
        - maximum_participants_constraint = null
        - minimum_active_participants_constraint = 0
        - maximum_active_participants_constraint = 0
        - authorized_role_changes = [(0,[2])]
* banned
    - role_index = 1
    - no capabilities
    - constraints
        - minimum_participants_constraint = 0

- maximum_participants_constraint = null
- minimum_active_participants_constraint = 0
- maximum_active_participants_constraint = 0
- authorized_role_changes = []

* ordinary_user
  - role_index = 2
  - authorized capabilities
    - canAddOwnClient
    - canAddSelf
    - canRemoveOwnClient
    - canRemoveSelf
    - canChangeOwnRole
    - canSendMessage
    - canReceiveMessage
    - canCopyMessage
    - canReportAbuse
    - canReactToMessage
    - canDeleteOwnReaction
    - canEditOwnMessage
    - canDeleteOwnMessage
    - canStartTopic
    - canReplyInTopic
    - canUploadImage
    - canUploadVideo

        - canUploadSound
        - canUploadAttachment
        - canDownloadImage
        - canDownloadVideo
        - canDownloadSound
        - canDownloadAttachment
        - canSendLink
        - canSendLinkPreview
        - canFollowLink
        - canCopyLink
        - canChangeOwnName
        - canChangeOwnPresence
        - canChangeOwnMood
        - canChangeOwnAvatar
    - constraints
        - minimum_participants_constraint = 0
        - maximum_participants_constraint = null
        - minimum_active_participants_constraint = 0
        - maximum_active_participants_constraint = null
        - authorized_role_changes = [(0,[2]), (2,[0])]
* group_admin
    - role_index = 3
    - authorized capabilities
        - (include all the capabilities authorized for an ordinary_user)

- o canAddParticipant
- o canRemoveParticipant
- o canBan
- o canUnBan
- o canKick
- o canChangeUserRole
- o canCreateJoinCode - reserved for future use
- o canDeleteOtherReaction
- o canDeleteOtherMessage
- o canEditOwnTopic
- o canEditOtherTopic
- o canChangeRoomDescription
- o canChangeRoomName
- o canChangeRoomAvatar
- o canChangeRoomSubject
- o canChangeRoomMood

- constraints
  - o minimum_participants_constraint = 1
  - o maximum_participants_constraint = null
  - o minimum_active_participants_constraint = 0
  - o maximum_active_participants_constraint = null
  - o authorized_role_changes = [(0,[1,2,3]), (1,[0,2,3]), (2,[0,1,3]), (3,[0,1,2])]

* super_admin
  - role_index = 4

- authorized capabilities
  - (include all the capabilities authorized for a group_admin)
  - canChangeRoomMembershipStyle
  - canChangeRoleDefinitions
  - canChangePreauthorizedUserList
  - canDestroyRoom
  - canSendMLSReinitProposal
- constraints
  - minimum_participants_constraint = 0
  - maximum_participants_constraint = null
  - minimum_active_participants_constraint = 0
  - maximum_active_participants_constraint = null
  - authorized_role_changes = [(0,[1,2,3,4]), (1,[0,2,3,4]), (2,[0,1,3,4]), (3,[0,1,2,4]), (4,[0,1,2,3])]

* policy_enforcer
  - role_index = 5
  - capabilities
    - (does not include any other capabilities)
    - canRemoveParticipant
    - canChangeUserRole
    - canBan
    - canUnban
    - canChangeRoleDefinitions
    - canChangePreauthorizedUserList
    - canChangeMlsOperationalPolicies

    - canDestroyRoom
    - canSendMLSReinitProposal
  - constraints
    - minimum_participants_constraint = 1
    - maximum_participants_constraint = 2
    - minimum_active_participants_constraint = 0
    - maximum_active_participants_constraint = 0
    - authorized_role_changes = [(0,[1]), (1,[0]), (2,[0,1]), (3,[0,1]), (4,[0,1])]
  - Notes: can remove a banned user from the list (cleanup) but not restore them

A.3. Moderated room

* no_role
  - role_index = 0
  - authorized capabilities
    - canUseJoinCode
  - constraints
    - minimum_participants_constraint = 0
    - maximum_participants_constraint = null
    - minimum_active_participants_constraint = 0
    - maximum_active_participants_constraint = null
    - authorized_role_changes = [(0,[2,3])]
* banned
  - role_index = 1
  - no capabilities

- constraints
  - minimum_participants_constraint = 0
  - maximum_participants_constraint = null
  - minimum_active_participants_constraint = 0
  - maximum_active_participants_constraint = 0
  - authorized_role_changes = []

* guest
  - role_index = 2
  - authorized capabilities
    - canRemoveSelf
    - canReceiveMessage
    - canCopyMessage
    - canReactToMessage
    - canDeleteOwnReaction
    - canFollowLink
    - canCopyLink
    - canDownloadImage
    - canDownloadVideo
    - canDownloadSound
  - constraints
    - minimum_participants_constraint = 0
    - maximum_participants_constraint = null
    - minimum_active_participants_constraint = 0
    - maximum_active_participants_constraint = null

- o authorized_role_changes = [(0,[2]), (2,[0])]

* attendee
  - role_index = 3
  - authorized capabilities
    - o (include all the capabilities authorized for a guest)
    - o canAddOwnClient
    - o canAddSelf
    - o canRemoveOwnClient
    - o canChangeOwnRole
    - o canReportAbuse
    - o canReplyInTopic
    - o canDownloadAttachment
    - o canChangeOwnName
    - o canChangeOwnPresence
    - o canChangeOwnAvatar
  - constraints
    - o minimum_participants_constraint = 0
    - o maximum_participants_constraint = null
    - o minimum_active_participants_constraint = 0
    - o maximum_active_participants_constraint = null
    - o authorized_role_changes = [(0,[3]), (3,[0])]
* speaker
  - role_index = 4
  - authorized capabilities

- (include all the capabilities authorized for a speaker)
- canSendMessage
- canEditOwnMessage
- canDeleteOwnMessage
- canStartTopic
- canUploadImage
- canUploadVideo
- canUploadSound
- canUploadAttachment
- canSendLink
- canSendLinkPreview

- constraints
  - minimum_participants_constraint = 0
  - maximum_participants_constraint = null
  - minimum_active_participants_constraint = 0
  - maximum_active_participants_constraint = null
  - authorized_role_changes = [(0,[4]), (4,[0])]

* moderator
  - role_index = 5
  - authorized capabilities
    - (include all the capabilities authorized for an ordinary_user)
    - canAddParticipant
    - canRemoveParticipant
    - canBan

        - canUnBan
        - canKick
        - canChangeUserRole
        - canCreateJoinCode - reserved for future use
        - canDeleteOtherReaction
        - canDeleteOtherMessage
        - canEditOwnTopic
        - canEditOtherTopic
        - canChangeRoomName
        - canChangeRoomAvatar
        - canChangeRoomSubject
        - canChangeRoomMood
    - constraints
        - minimum_participants_constraint = 1
        - maximum_participants_constraint = null
        - minimum_active_participants_constraint = 0
        - maximum_active_participants_constraint = null
        - authorized_role_changes = [(0,[1,2,3,4,5]), (1,[0,2,3,4,5]), (2,[0,1,3,4,5]), (3,[0,1,2,4,5]), (4,[0,1,2,3,5]), (5,[0,1,2,3,4])]

* super_admin
    - role_index = 6
    - authorized capabilities
        - (include all the capabilities authorized for a moderator)
        - canChangeRoomDescription

- o canChangeRoomMembershipStyle
- o canChangeRoleDefinitions
- o canChangePreauthorizedUserList
- o canDestroyRoom
- o canSendMLSReinitProposal

- constraints
  - o minimum_participants_constraint = 0
  - o maximum_participants_constraint = null
  - o minimum_active_participants_constraint = 0
  - o maximum_active_participants_constraint = null
  - o authorized_role_changes = [(0,[1,2,3,4,5,6]), (1,[0,2,3,4,5,6]), (2,[0,1,3,4,5,6]), (3,[0,1,2,4,5,6]), (4,[0,1,2,3,5,6]), (5,[0,1,2,3,4,6]), (6,[0,1,2,3,4,5])]

* policy_enforcer
  - role_index = 7
  - capabilities
    - o (does not include any other capabilities)
    - o canRemoveParticipant
    - o canChangeUserRole
    - o canBan
    - o canUnban
    - o canChangeRoleDefinitions
    - o canChangePreauthorizedUserList
    - o canChangeMlsOperationalPolicies
    - o canDestroyRoom

    - o canSendMLSReinitProposal

  - constraints

    - o minimum_participants_constraint = 1

    - o maximum_participants_constraint = 2

    - o minimum_active_participants_constraint = 0

    - o maximum_active_participants_constraint = 0

    - o authorized_role_changes = [(0,[1]), (1,[0]), (2,[0,1]), (3,[0,1]), (4,[0,1]), (5, [0,1]), (6, [0,1])]

  - Notes: can remove a banned user from the list (cleanup) but not restore them

A.4. Multi-organization administered room

In this example room policy, Alice from organization A is a super admin. There are per organization user and admin roles for orgs A, B, and C. Organizational admins can only move users to and from their org user role, their org admin role, the no_role; and can ban (but not unban) their own org users. The non-host orgs do not have the canChangeOwnRole and canAddSelf, and are limited to 3 admins per org.

* no_role

  - role_index = 0

  - no capabilities

  - constraints

    - o minimum_participants_constraint = 0

    - o maximum_participants_constraint = null

    - o minimum_active_participants_constraint = 0

    - o maximum_active_participants_constraint = 0

    - o authorized_role_changes = []

* banned

- role_index = 1
- no capabilities
- constraints
  - minimum_participants_constraint = 0
  - maximum_participants_constraint = null
  - minimum_active_participants_constraint = 0
  - maximum_active_participants_constraint = 0
  - authorized_role_changes = []

* org_a_user
  - role_index = 2
  - authorized capabilities
    - (all capabilities of org_b_user)
    - canChangeOwnRole
    - canAddSelf
  - constraints
    - minimum_participants_constraint = 0
    - maximum_participants_constraint = null
    - minimum_active_participants_constraint = 0
    - maximum_active_participants_constraint = null
    - authorized_role_changes = [(0,[2]), (2,[0])]

* org_b_user
  - role_index = 3
  - authorized capabilities
    - canRemoveSelf

    - o canAddOwnClient
    - o canRemoveOwnClient
    - o canSendMessage
    - o canReceiveMessage
    - o canCopyMessage
    - o canReportAbuse
    - o canReplyInTopic
    - o canReactToMessage
    - o canDeleteOwnReaction
    - o canEditOwnMessage
    - o canSendLink
    - o canSendLinkPreview
    - o canFollowLink
    - o canCopyLink
    - o canDownloadImage
    - o canDownloadVideo
    - o canDownloadSound
    - o canDownloadAttachment
    - o canChangeOwnName
    - o canChangeOwnPresence
    - o canChangeOwnAvatar
  - constraints
    - o minimum_participants_constraint = 0
    - o maximum_participants_constraint = null

        - minimum_active_participants_constraint = 0
        - maximum_active_participants_constraint = null
        - authorized_role_changes = [(0,[3]), (3,[0])]
    * org_c_user
        - role_index = 4
        - authorized capabilities
            - (same capabilities as org_b_user)
        - constraints
            - minimum_participants_constraint = 0
            - maximum_participants_constraint = null
            - minimum_active_participants_constraint = 0
            - maximum_active_participants_constraint = null
            - authorized_role_changes = [(0,[4]), (4,[0])]
    * org_a_admin
        - role_index = 5
        - authorized capabilities
            - (all capabilities of org_b_admin)
            - canChangeOwnRole
            - canAddSelf
        - constraints
            - minimum_participants_constraint = 0
            - maximum_participants_constraint = null
            - minimum_active_participants_constraint = 0
            - maximum_active_participants_constraint = null

        - authorized_role_changes = [(0,[2,5]), (2,[0,1,5]), (5,[0,1,2])]

* org_b_admin
    - role_index = 6
    - authorized capabilities
        - (all capabilities of org_b_user)
        - canDeleteOwnMessage
        - canStartTopic
        - canUploadImage
        - canUploadVideo
        - canUploadSound
        - canUploadAttachment
        - canAddParticipant
        - canRemoveParticipant
        - canBan
        - canKick
        - canChangeUserRole
    - constraints
        - minimum_participants_constraint = 1
        - maximum_participants_constraint = 3
        - minimum_active_participants_constraint = 1
        - maximum_active_participants_constraint = null
        - authorized_role_changes = [(0,[3,6]), (3,[0,1,6]), (6,[0,1,3])]

* org_c_admin

- role_index = 7
- authorized capabilities
  - (all capabilities of org_b_admin)
- constraints
  - minimum_participants_constraint = 1
  - maximum_participants_constraint = 3
  - minimum_active_participants_constraint = 1
  - maximum_active_participants_constraint = null
  - authorized_role_changes = [(0,[4,7]), (4,[0,1,7]), (7,[0,1,4])]

* super_admin
  - role_index = 8
  - authorized capabilities
    - (include all the capabilities authorized for org_a_admin)
    - canUnBan
    - canDeleteOtherReaction
    - canDeleteOtherMessage
    - canEditOwnTopic
    - canEditOtherTopic
    - canChangeRoomDescription
    - canChangeRoomName
    - canChangeRoomAvatar
    - canChangeRoomSubject
    - canChangeRoomMood
    - canChangeRoomMembershipStyle

        - canChangeRoleDefinitions
        - canChangePreauthorizedUserList
        - canDestroyRoom
        - canSendMLSReinitProposal
    - constraints
        - minimum_participants_constraint = 1
        - maximum_participants_constraint = null
        - minimum_active_participants_constraint = 1
        - maximum_active_participants_constraint = null
        - authorized_role_changes = [(0,[1,2,3,4,5,6,7,8]), (1,[0,2,3,4,5,6,7,8]), (2,[0,1,5,8]), (3,[0,1,6]), (4,[0,1,7]), (5,[0,1,2,8]), (6,[0,1,3]), (7,[0,1,4]), (8,[0,1,2,5])]
* policy_enforcer
    - role_index = 9
    - capabilities
        - (does not include any other capabilities)
        - canRemoveParticipant
        - canChangeUserRole
        - canBan
        - canUnban
        - canChangeRoleDefinitions
        - canChangePreauthorizedUserList
        - canChangeMlsOperationalPolicies
        - canDestroyRoom
        - canSendMLSReinitProposal

- constraints
  - minimum_participants_constraint = 1
  - maximum_participants_constraint = 2
  - minimum_active_participants_constraint = 0
  - maximum_active_participants_constraint = 0
  - authorized_role_changes = [(0,[1]), (1,[0]), (3,[0,1]), (4,[0,1]), (5,[0,1]), (6,[0,1]), (7,[0,1]), (8,[0,1])]
- Notes: can remove a banned user from the list (cleanup) but not restore them

Acknowledgments

TODO acknowledge.

Author’s Address

Rohan Mahy
Rohan Mahy Consulting Service
Email: rohan.ietf@gmail.com